

Laboratorio 7 — Registro Esami con struct, enum e File Binario

Live Demo — career_bin.c

Università di Catania — Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Programmazione 1 e Laboratorio (Canale O-Z)

Descrizione del Programma

Si scriva un programma in linguaggio C, chiamato `career_bin.c`, che gestisca un archivio binario degli esami sostenuti in un corso di laurea. Ogni esame è rappresentato da una struttura contenente informazioni su materia, codice, crediti, voto e stato dell'esame, e viene memorizzato su disco in formato binario.

Il programma deve essere invocato da riga di comando con una delle seguenti modalità operative:

```
./career_bin new carriera.bin
./career_bin load carriera.bin
./career_bin stats carriera.bin
```

Il file binario `carriera.bin` non contiene array paralleli ma un numero di esami seguito da un array di strutture omogenee, scritte con `fwrite` e lette con `fread`.

È richiesto l'uso delle seguenti definizioni di tipo:

```
typedef enum {
    IN_CORSO,
    SUPERATO,
    BOCCIATO
} StatoEsame;

typedef struct {
    char nome[50];
    char codice[10];
    int  cfu;
    int  voto;
    StatoEsame stato;
} Esame;
```

In memoria, il programma lavora su un array dinamico di `Esame` allocato nello heap tramite `malloc` e deallocato con `free`.

Requisiti di Implementazione

Il programma deve essere strutturato nei seguenti punti:

PUNTO A: Verificare che il numero di argomenti passati da riga di comando sia esattamente 3, ovvero il nome dell'eseguibile, la modalità (**new**, **load**, **stats**) e il nome del file binario. Se il numero di argomenti è diverso da 3, stampare:

```
Errore: uso corretto ./career_bin [new|load|stats] file.bin
```

e terminare con exit status 1. Se la modalità specificata non è una tra quelle previste, stampare:

```
Errore: modalita non riconosciuta.
```

e terminare con exit status 1.

PUNTO B: Definire i tipi **StatoEsame** ed **Esame** come indicato nella descrizione e implementare una funzione di parsing di una riga testuale nel formato **nome;codice;cfu;voto;stato**. La funzione deve validare che nome e codice siano non vuoti, che i crediti siano strettamente positivi, che il voto sia compreso tra 18 e 30 solo se lo stato è **SUPERATO**, e che gli stati ammessi siano solo **IN_CORSO**, **SUPERATO** e **BOCCIATO**. In caso di riga non valida, stampare:

```
Errore: riga esame non valida.
```

e terminare con exit status 1.

PUNTO C: In modalità **new**, leggere da standard input una sequenza di righe con **fgets** fino a EOF, creare dinamicamente un array di **Esame** e salvare gli esami nel file binario **carriera.bin**.

- Se l'apertura del file fallisce, stampare:

```
Errore: impossibile aprire il file binario per scrittura.
```

e terminare con exit status 1.

- Al termine, scrivere prima il numero di esami come **int** e poi l'intero array con **fwrite**.
- In caso di successo, stampare:

```
Archivio creato con successo.
```

e terminare con exit status 0.

PUNTO D: In modalità `load`, aprire il file binario in lettura, leggere il numero di esami e l'intero array in memoria, quindi stampare l'elenco con l'intestazione:

Elenco esami:

Ogni esame va stampato in una riga nel formato:

Nome (CODICE) - CFU: X - Voto: Y - Stato: STATO

- Se il file non può essere aperto, stampare:

Errore: impossibile aprire il file binario per lettura.

- Se il file binario è corrotto, stampare:

Errore: file binario corrotto.

- Se non ci sono esami, stampare:

Elenco esami:

Nessun esame presente.

PUNTO E: In modalità `stats`, leggere gli esami come nel punto precedente e calcolare numero totale di esami, numero di esami per ciascuno stato, crediti acquisiti, media pesata sui soli esami `SUPERATO`, esame con voto massimo e minimo tra i soli `SUPERATO`.

- La media pesata è data da:

$$\text{media pesata} = \frac{\sum (\text{voto}_i \cdot \text{cfu}_i)}{\sum \text{cfu}_i}$$

- Stampare un riepilogo nel formato indicato negli esempi di output di controllo.
- Se non esiste alcun esame `SUPERATO`, stampare N/D al posto di media, migliore e peggiore.
- Il programma deve terminare con exit status 0 dopo aver deallocato la memoria.

Output di Controllo (Esempi di Esecuzione)

Caso 1 — Creazione dell'archivio e lettura successiva

Comando:

```
./career_bin new carriera.bin
```

Input da standard input:

```
Programmazione 1;INF101;9;30;SUPERATO  
Analisi Matematica;MAT101;12;27;SUPERATO  
Fisica;FIS101;6;0;IN_CORSO
```

Il programma deve stampare a schermo:

Archivio creato con successo.

(Exit code atteso: 0)

Comando:

```
./career_bin load carriera.bin
```

Il programma deve stampare a schermo:

```
Elenco esami:  
Programmazione 1 (INF101) - CFU: 9 - Voto: 30 - Stato: SUPERATO  
Analisi Matematica (MAT101) - CFU: 12 - Voto: 27 - Stato: SUPERATO  
Fisica (FIS101) - CFU: 6 - Voto: 0 - Stato: IN_CORSO
```

(Exit code atteso: 0)

Caso 2 — Statistiche sulla carriera

Comando:

```
./career_bin stats carriera.bin
```

Il programma deve stampare a schermo:

```
Numero esami: 3  
IN_CORSO: 1  
SUPERATO: 2  
BOCCIATO: 0  
Crediti acquisiti: 21  
Media pesata: 28.29  
Esame migliore: Programmazione 1 (INF101) - Voto: 30 - CFU: 9  
Esame peggiore: Analisi Matematica (MAT101) - Voto: 27 - CFU: 12
```

(Exit code atteso: 0)

Caso 3 — File binario corrotto

Se il file binario è incompleto o contiene un numero di esami non coerente con la dimensione del file, il programma deve stampare:

Errore: file binario corrotto.

(Exit code atteso: 1)